

CLAIMS

1. A microprocessor and support software, comprising:
an instruction pipeline designed to execute instructions of an instruction set, control-transfer instructions of the instructions being instructions defined to transfer execution control of a computer from a source instruction to a destination instruction, control-flow instructions of the instruction set being classified into a relatively small plurality of classes relative to the number of instruction opcodes executable by the instruction pipeline, most divisions in the classification being based on a static encoding of control-flow instructions executed, with at most minor divisions in the classification being based on dynamic or data-dependent execution behavior;
a storage register designed to store, and updating circuitry active during execution of a program on the microprocessor, designed to record into the storage register, as part of the execution of control-flow instructions of the instruction set and without software intervention, a value reflecting the class of a control-flow instruction recently executed by the pipeline;
software programmed to adjust the storage contents of the computer to reestablish in the context of the destination instruction a context logically equivalent to the context of the computer at the time of the control transfer instruction, the adjustment being determined, at least in part, by a classification of the control-transfer instruction; and
invoking circuitry to invoke the software before executing the destination instruction of at least some of the control transfer instructions.

2. The microprocessor of claim 1, wherein:
at least three-quarters of the classification divisions are defined solely by static encoding of instructions executed;
at least some of the classification divisions are defined by the opcode of the instructions;
and
at least some of the classification divisions are defined by immediate values of instructions.

3. The method of claim 1, wherein:

2 at least four of the classification divisions distinguish different classes of jump
3 instructions emitted by a typical compiler for the computer primarily to effect intra-subprogram
4 transfers; and

5 at least half of the classification divisions distinguish classes of control transfers that
6 either (a) induce a context switch, (b) are emitted by a typical compiler for the computer
7 primarily to effect a control transfer between subprograms, or (c) effect argument passing or
8 return as a side effect of the control transfer.

1 4. The method of claim 1, wherein:

2 at least some of the immediate values defining the classification divisions are branch
3 displacements; and

4 at least some of the immediate values defining the classification divisions are immediate
5 values having no effect on the execution of the instructions in which they are encoded, except to
6 update the storage register.

7 5. The method of claim 1, wherein at least some instructions of the computer are
8 assigned to a classification division ignored by the circuitry, execution of instructions in this
9 ignored classification leaving intact the previous contents of the storage register.

10 6. A method, comprising:

11 classifying control-flow instructions of a computer instruction set into a plurality of
12 classes; and

13 during execution of a program on a computer, as part of the execution of instructions of
14 the instruction set, updating a record of the class of the classified control-flow instruction most
15 recently executed.

16 7. The method of claim 6, further comprising:

17 detecting when the computer's execution flows from code observing a first data storage
18 convention to code observing a second data storage convention, the record recording a class of
19 instruction effecting the transitional execution flow; and

5 in response to the detecting, altering the data storage content of the computer to create a
6 program context under the second data storage convention that is logically equivalent to a pre-
7 alteration program context under the first data storage convention, the altering process being
8 determined, at least in part, by the instruction classification record.

1 8. The method of claim 7, wherein:
2 one of the two data storage conventions is a register-based calling convention, and the
3 other data storage convention is a memory stack-based calling convention.

1 9. The method of claim 8, wherein:
2 at least four of the classification divisions distinguish different classes of jump
3 instructions emitted by a typical compiler for the computer primarily to effect intra-subprogram
4 transfers; and

5 at least half of the classification divisions distinguish classes of control transfers that
6 either (a) induce a context switch, (b) are emitted by a typical compiler for the computer
7 primarily to effect a control transfer between subprograms, or (c) effect argument passing or
8 return as a side effect of the control transfer.

1 10. The method of claim 8, wherein:
2 in some of the control-flow instructions, the classification is dynamically determined
3 based on a full/empty status of a register.

1 11. The method of claim 6, wherein:
2 in some of the control-flow instructions, the classification is encoded as an immediate
3 value of instructions, the immediate value having no effect on the execution of the instruction in
4 which it is encoded, except to update the class record; and
5 in some of the control-flow instructions, the classification is statically determined by the
6 opcode of the instructions.

1 12. The method of claim 6, wherein:

2 at least four of the classification divisions distinguish different classes of jump
3 instructions emitted by a typical compiler for the computer primarily to effect intra-subprogram
4 transfers; and

5 at least half of the classification divisions distinguish classes of control transfers that
6 either (a) induce a context switch, (b) are emitted by a typical compiler for the computer
7 primarily to effect a control transfer between subprograms, or (c) effect argument passing or
8 return as a side effect of the control transfer.

1 13. A method, comprising:
2 executing a control-transfer instruction that transfers execution control of a computer
3 from a first execution context to a destination instruction for execution in a second execution
4 context;

5 before executing the destination instruction, reconfiguring the storage state of the
6 computer to reestablish under the second execution context the logical state of the computer as
7 interpreted under the first execution context;

8 the reconfiguring being determined, at least in part, by a classification of the control-
9 transfer instruction.

10 14. The method of claim 13, wherein the destination instruction is an entry point to an
11 off-the-shelf binary for an operating system coded in an instruction set non-native to the
12 computer, the non-native instruction set providing access to a reduced subset of the resources of
13 the computer.

1 15. The method of claim 13, wherein:
2 at least four of the classification divisions distinguish different classes of jump
3 instructions emitted by a typical compiler for the computer primarily to effect intra-subprogram
4 transfers; and
5 at least half of the classification divisions distinguish classes of control transfers that
6 either (a) induce a context switch, (b) are emitted by a typical compiler for the computer

7 primarily to effect a control transfer between subprograms, or (c) effect argument passing or
8 return as a side effect of the control transfer.

1 16. The method of claim 13, wherein:
2 in some of the control-flow instructions, the classification is encoded as an immediate
3 value of instructions, the immediate value having no effect on the execution of the instruction in
4 which it is encoded, except to update the class record.

1 17. The method of claim 13, wherein:
2 in some of the control-flow instructions, the classification is statically determined by the
3 opcode of the instructions; and
4 in other of the control-flow instructions, the classification is dynamically determined with
5 reference to a state of a processor register and/or data register of the computer.

6 18. The method of claim 17, wherein:
7 for some of the control-flow instructions, the dynamic classification is determined based
8 on a full/empty status of a register.

9 19. The method of claim 13:
10 wherein one of the two data storage conventions is a register-based calling convention,
11 and the other data storage convention is a memory stack-based calling convention;
and further comprising:
detecting when the computer's execution flows from code observing the register-based
calling convention to code observing the memory stack-based calling convention, the record
recording the classification of the instruction effecting the transitional execution flow; and
in response to the detecting, altering the data storage content of the computer to create a
program context under the memory stack-based calling convention that is logically equivalent to
a pre-alteration program context under the register-based calling convention convention, the
altering process being determined, at least in part, by the instruction classification record.

1 20. The method of claim 19, wherein:
2 at least four of the classification divisions distinguish different classes of jump
3 instructions emitted by a typical compiler for the computer primarily to effect intra-subprogram
4 transfers; and
5 at least half of the classification divisions distinguish classes of control transfers that
6 either (a) induce a context switch, (b) are emitted by a typical compiler for the computer
7 primarily to effect a control transfer between subprograms, or (c) effect argument passing or
8 return as a side effect of the control transfer.

1 21. The method of claim 19, wherein:
2 in some of the control-flow instructions, the classification is encoded as an immediate
3 value of the instructions, the immediate value having no effect on the execution of the instruction
4 in which it is encoded, except to update the class record.

1 22. The method of claim 19, wherein:
2 in some of the control-flow instructions, the classification is statically determined by the
3 opcode of the instructions; and
4 in other of the control-flow instructions, the classification is dynamically determined
5 based on a full/empty status of a register.

1 23. The method of claim 19, wherein:
2 the rearranging is performed by an exception handler, the handler being selected by an
3 exception vector based at least in part on the instruction classification record.

1 24. A microprocessor, comprising:
2 an instruction pipeline designed to execute instructions of an instruction set, the
3 instructions of the instruction set being classified into a relatively small number of classes
4 relative to the number of instruction opcodes executable by the instruction pipeline, most
5 divisions in the classification being based on a static encoding of instructions executed, with at

6 most minor divisions in the classification being based on dynamic or data-dependent execution
7 behavior;

8 a storage register designed to store, and circuitry designed to record without software
9 intervention into the storage register, a value reflecting the class of an instruction recently
10 executed by the pipeline.

1 25. The microprocessor of claim 24, wherein at least three-quarters of the
2 classification divisions are defined solely by static encoding of instructions executed.

1 26. The microprocessor of claim 25, wherein:
2 some of the classification divisions are defined by the opcode of the instructions; and
3 some of the classification divisions are defined by immediate values of instructions.

1 27. The microprocessor of claim 26, wherein:
2 at least four of the classification divisions distinguish different classes of jump
3 instructions emitted by a typical compiler for the computer primarily to effect intra-subprogram
4 transfers; and

5 at least half of the classification divisions distinguish classes of control transfers that
6 either (a) induce a context switch, (b) are emitted by a typical compiler for the computer
7 primarily to effect a control transfer between subprograms, or (c) effect argument passing or
8 return as a side effect of the control transfer.

1 28. The microprocessor of claim 26, wherein at least some of the immediate values
2 defining the classification divisions are branch displacements.

1 29. The microprocessor of claim 26, wherein at least some of the immediate values
2 defining the classification divisions are immediate values having no effect on the execution of
3 the instructions in which they are encoded, except to update the storage register.

1 30. The microprocessor of claim 24, wherein:

2 at least some of the classification divisions are determined by dynamic reference to a
3 state of architecturally-visible processor registers and/or general purpose registers of the
4 computer.

1 31. The microprocessor of claim 30, wherein the state of the register is a state
2 maintained distinct from a data content of the register.

1 32. The microprocessor of claim 24, wherein at least some instructions of the
2 computer are assigned to a classification division ignored by the circuitry, execution of
3 instructions in this ignored classification leaving intact the previous contents of the storage
4 register.

1 33. The microprocessor of claim 24, wherein the number of classes is at most 32.

1 34. The microprocessor of claim 24, further comprising hardware or software
2 designed to use the content of the storage register to annotate a descriptor in a profile of the
3 execution of a program running on the microprocessor.
4